MUSIC TRANSFORMATION IN CELLULAR SPACES TRIGGERED BY WALSH FUNCTIONS

Patricio da Silva
Spectrum Press
P.O. Box 50786
Los Angeles, CA 90050
USA
pdasilva@spectrumpress.com

Adolfo Maia Jr.
IMECC and NICS, UNICAMP,
13.081-970 - Campinas (SP), Brazil.
Future Music Lab
University of Plymouth
adolfo@nics.unicamp.br

ABSTRACT

This paper introduces a new compositional process based on transformations of previously existing material by segmentation of information located in a 2-dimensional cellular-space, the use of Walsh Functions as triggers, and recombinancy by cyclic transposition. The process can produce a wide spectrum of results depending on the size of segmentation, the choice of the Walsh functions, the operation(s) used, and the parameters entered for each operation. These results can range from a simple variation of the input, to an output, holding little or no resemblance with the original.

1. INTRODUCTION

As many dynamical systems, certain compositional processes can be decomposed into components where the state of the system as a whole results from the transition of state of its individual components. The concept of compositional processes as mechanisms can be documented back to Guido dÁrezzo (ca. 991- ca. 1028) and his *Micrologus* (1025-26), where dÁrezzo described an algorithm capable of producing a melodic line from any input text [1]. This paper focus on the analysis of the particular device used in two distinct compositions. In a couple of manuscripts of Johann Sebastian Bachs (1685-1750), *Prelude I* in C Major (BWV 846a), and *Prelude II* in C minor (BWV 847/1) (from the Well Tempered Clavier I), shown in pages 16-17 of the reference [1], in order to speed up the process of copying the actual music content, only a few initial measures were fully notated as in a modern performance score. In both manuscripts, once we have a few measures written out (a template), the remaining of the piece is described solely as a sequence of harmonic structures for which is left implied that the performer knows how to go on with it. For a performer, its readily apparent how certain configurations of contour repeat in cycles. The notes change, but the choreography of the keyboardists fingers remains identical. For each harmony, the melodic pattern functions as a list of instructions to how articulate notes in time. To implement Bachs mechanism we need two inputs: a melodic pattern, and an array of harmonies (listed from lowest to highest pitches). For each iteration of the melodic pattern the system combines it with a different harmony until exhausting all harmonies. A melodic pattern is a sequence of numbers, where, at discrete time-steps, each number functions as pointer to the note in the harmony with the same index-number. Preludes I and II share the process by which the different notes that constitute a harmony are articulated in time. The melodic pattern that serves as template in prelude I points to the notes indexed as $[01234234 - 01234234]$ while prelude II uses the following pointers:$[03231323 - 03231323]$(left-hand), $[765664656 - 765664656]$ (right-hand). From these Bach's examples we understand how melodic patterns, here coded as numerical sequences, function as triggers in time selecting certain notes from the current harmony. Our work expands the concept of composition as the result of independent components, and presents yet a new algorithm for the transformation of previously existing material.

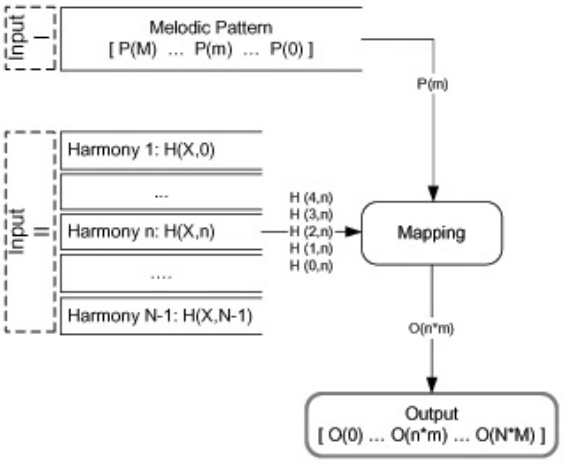

Figure 1. Bach's scheme of composition.

We describe, in the next sections, a modular process of transformation supported by the encoding, selection and segmentation of a working area in a Cellular Space (CS), which can be the entire CS as well, and the use of Walsh Functions as triggers for specified transformations. For

the sake of simplicity, we restrict our description to a particular transformation, namely, *transposition* in CS considered as a closed (in the pitch-layers direction) cyclic space, that promotes a kind of recombination of the information input.

Recombination is understood here diferently than in Cope's work with Experiments in Musical Intelligence[5]. For Cope, recombination is operated on a data-base of works by a given composer. Each work in the database is divided into segments (typically one or two measures long), and each segment catalogued according to its function in the grammar SPEAC (an Augmented Transition Network). New works can be composed by re-combining segments, originally from different works, in a new hierarchically and stylistically plausible context. While in Cope's work the idea is to replicate the style(s) present in a database, we aim at the transformation of a given input, which can be as short as one note, or as long as a complete composition, by manipulating music data directly at the lowest level of MIDI based music representation. The resulting degree of transformation can range from a slight variation of the input, to an output which has no recognizable link to the original. This implies that, in general, we are not able to predict the ouput, unless for very simple and short inputs.

2. CELLULAR SPACE

We present in this section, for the sake of completeness, a short review of Cellular Space, and how it can be used in a a variety of contexts. A more complete presentation can be found in [2, 3] .

2.1. Definition

Our approach on encoding music information on Cellular Space can provide the ground-level representation for many different systems. It was originally developed for string rewriting, but its simplicity makes it a flexible platform suitable for many different applications including, among others, substitution systems, recombinancy, and cellular automata. Roughly speaking, given a music material input, a Cellular Space is a matrix of empty cells associated to it. Formally, a Cellular Space is a 2-dimensional space $W = \{(n, p), \quad \text{with} \quad 0 \leq t \leq N, \text{and} \quad P_0 \leq p \leq P_r\}$, where n= cell-index, p=pitch-layer, P_0 is the lowest pitch-layer, and P_r the highest, and N is the total number of cells per pitch-layer. In this model, cells are windows of time set to a time-duration unit we named *cell-size* which we denote by s. So, the total time can be written as $T = N \times s$. In addition the on-time the n-th cell can be written as $t = n \times s$.

2.2. Representation

Cellular Space is internally represented as an array of lists (a list of lists), where each of the contained lists (a sub-list) accounts for a pitch-layer (in the input, the set of events sharing the same MIDI-note). The sequence of sub-lists is zero-indexed (the first sub-list numbered 0, the second numbered 1, and so forth), and organized from the lowest to the highest pitch-layer they represent.

As defined above, each pitch-layer, contains exactly the same number of cells and these cells (represented as binary digits) are also zero-indexed. The sequence of cells samples the time axis by cell-size up to the maximal time T. The $r + 1$ pitch-layers in Cellular Space forms the pitch-range covered in the input and it is calculated as the difference between the highest and the lowest sounding notes. The user can, however, override these values by replacing them with others that extend the default range within the limits of MIDI systems (note values from 0 to 127 only). The encoding algorithm produces a unique cellular space per MIDI-channel, outputting as many parallel spaces as the number of the different channels found in the input events. Typically a MIDI-channel can carry multiple instruments, however, to simplify, the input conforms here to a exclusive assignment of a channel per instrument, so that, each channel/instrument gets it's own cellular-space. Given that no pitch-layer has any particular information regarding which note-number it represents, the system registers in a variable the lowest note-number (*lowest-note*) being represented. This variable is used as a reference point, where the first sub-list (index 0) represents the activity for the lowest note in the input, and each subsequent sub-list follows a note-number increment by 1 (a minor-second up).

Cellular Space can be graphically represented as a matrix of regularly spaced cells, a two-dimensional lattice of N identical cells as in Example 2. Each row of cells represents a pitch-layer, and columns are windows of time represented, where each cell has an on-time and a default duration.

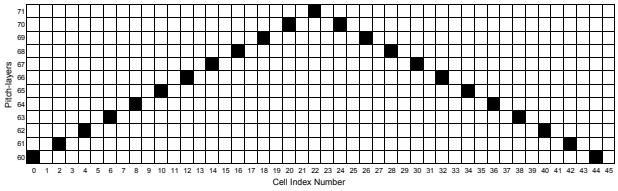

Figure 2. An example of Cellular encoding (graphical representation) of a sequence of events, here a chromatic scale.

3. SEGMENTATION AND TRIGGERING BY WALSH FUNCTIONS

Our approach is based on two operations, namely, *segmentation of the pitch-layers*, and afterwards *triggering by Walsh Functions*, which can be chosen independently one from another. This allows the system a great flexibility to perform transformations on a given material. In a segmentation process, a pitch-layer is truncated (i.e., sliced) by a constant length (segment-size) defined by the user. Clearly, it may occur that dividing the length of each pitch-layer (i.e., the number of cells) by segment-size does

$$\left[\begin{array}{ccc|ccc|ccc|ccc|ccc|c} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{array}\right]$$

Figure 3. A segmented Cellular Space

not result in an integer, in which case, the last segment includes the remaining cells.

As described above, a CS could be viewed as a matrix where each row is a pitch-layer. With this representation Example 3 shows a CS sliced by segments of size 3.

For each pitch-layer, consecutive values from the selected Walsh function, in this case (1 -1 1 -1), are assigned, one per segment, to the respective segments. The distribution of values from the designated Walsh function is processed as loop, so that in those situations where the number of segments exceeds the size of the Walsh function, the algorithm returns to the beginning of the selected function, repeating this process until all segments have been exhausted. In our code the location of a segment is parametrized by the variables *segment-index* corresponding to the segment's on-time and its *pitch-layer* index. So all the basic instructions are located in substrings of the form $(walsh-value, segment, segment-index, pitch-layer \quad index)$ which are nested in a convenient way. For example, $(1(001)(20))$ is to be read: walsh-value = 1, segment = (0 0 1), segment-index = 2, pitch-layer index = 0.

Each value of a Walsh function works as a trigger calling the operation associated with it. For example, a typical instruction could be: if Walsh value is equal to -1, then do the operation Transposition in the segment with index n by -3. In each iteration of the system all pitch-layers are given triggers from same Walsh function. Further details about Walsh Functions can be found in the Appendix section.

In Figure 4 we present a diagram of our implementation of the algorithm and a figure how the Walsh function modify a binary code.

4. ANALYSIS OF SOME EXPERIMENTS

Triggering operations with Walsh functions can produce an endless number of outputs with only slight variations in the parameters' values. The results from the triggering transformations depend of too many variables to be easily predicted. The initial results show that the size of segments plays a major role in the impact of the transformation. The smaller are the segments, the larger the number of triggers distributed along the selected material. When a trigger calls a certain operation on a segment, the results will vary according to the actual configuration of 1's and 0's in the pitch-layers. With the operation of transposition, as previously described, we achieve the re-allocation of cells as arranged in segments. For example, the operation may result in inserting a segment of digits that corresponded to part of a sustained sound, in a pitch-layer that had previously no articulations to play. This may also result in the activation of new pitches, the creation of new-voices, new articulations, and poly-rhythms. Below we show some simple examples and the used parameters' values, as well as the correspondent scores showing both the input and output for comparison. Sound examples showing transformations in works of some composers are available in the internet at http://www.spectrumpress.com/dasilva/walsh-examples.html.

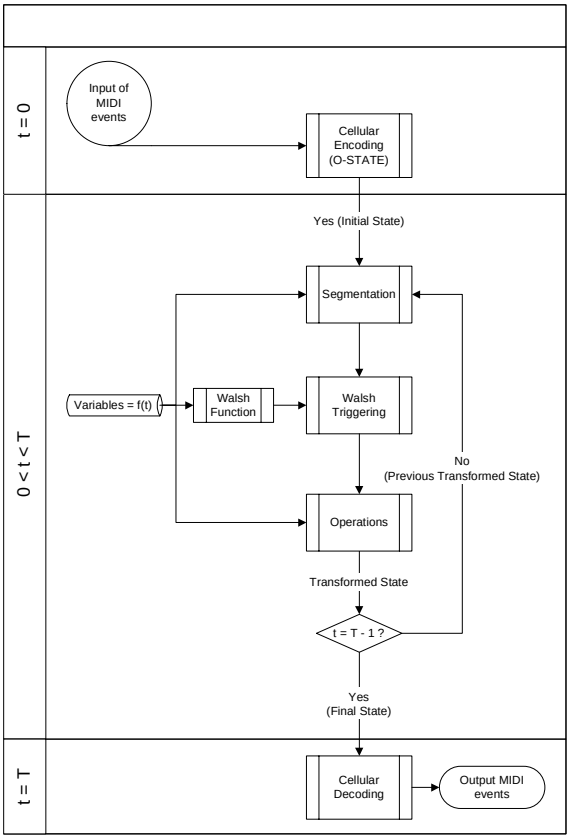

Figure 4. Diagram of the Algorithm Implementation.

Example 1: In this example all notes have the same duration. We've chosen a segmentation length igual to 2, which is proportional to the constant duration of the notes. It is easy to see that only the pitches were transposed but no rhythmic modification occured. See figure 4.

- Segmentation length $= 2$
- Walsh function : $(1, 1, -1, -1, 1, 1, -1, -1)$
- Trigger $= -1$
- Transposition $= -2$

Example 2: The re-allocation of segments with binary numbers 1s (in general, associated with the production of sound) causes the appearance of new voices and increasing rhythmic complexity. See figure 5.

- Segmentation length $= 3$
- Walsh function : $(1, -1, 1, -1, 1, -1, 1, -1)$

- Trigger $= -1$
- Transposition $= -4$

Example 3: In this example we've just added in the input score a whole-note (B) in the first measure of the lower voice (bass clef). In addition to a higher rhythmic complexity, as in the previous example, note the appearance of vertical structures.

- Segmentation length $= 3$
- Walsh function : $(1, -1, 1, -1, 1, -1, 1, -1)$
- Trigger $= -1$
- Transposition $= -4$

Example 4:

In this example, we show the progressive change from the original material through a sucession of transformations, each one of them with their own parameters, and effects. When comparing with example 1, we see that the rhythmic uniformity of the input left little opportunity to radical rhythmic changes. In the last iteration of the algorith, we used a segmentation by 1 cell, where trigger -1 called the operation of transposition by -3. However,it so happens that with this particular input, all segments with trigger -1 included only the digit 0, so the re-allocation of segments resulted in a identical cellular space.

4a) First change

- Segmentation length $= 4$
- Walsh function : $(1, 1, -1, -1, -1, -1, 1, 1)$
- Trigger $= 1$
- Transposition $= 2$

4b) Second change

- Segmentation length $= 2$
- Walsh function : $(1, -1, 1, -1, 1, -1, 1, -1)$
- Trigger $= -1$
- Transposition $= -1$

4c) Third change

- Segmentation length $= 2$
- Walsh function : $(1, -1, 1, -1, 1, -1, 1, -1)$
- Trigger $= -1$
- Transposition $= -1$

4d) Fourth change

- Segmentation length $= 1$
- Walsh function : $(1, 1, -1, -1, -1, -1, 1, 1)$
- Trigger $= -1$
- Transposition $= -3$

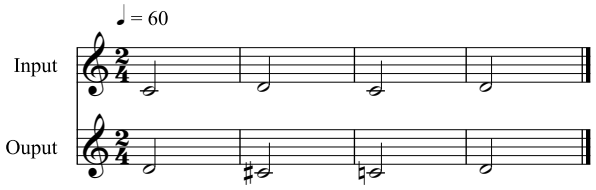

Figure 5. Tranformation of long notes .

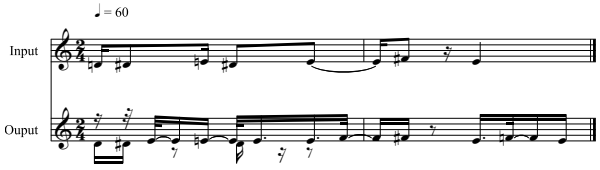

Figure 6. Rhythm diversity and new voices in a transformed melodic line.

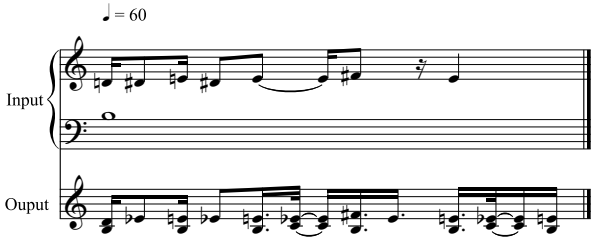

Figure 7. Rhythm diversity and new chords formation.

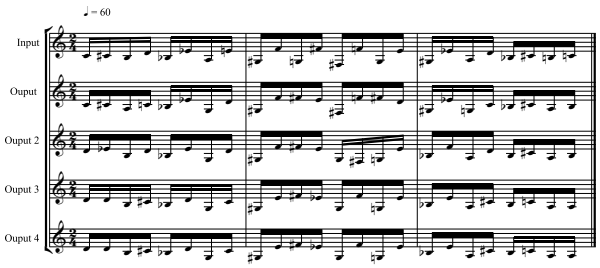

Figure 8. Four examples showing iterated transformations of a music material.

4.1. Conclusion and Perspectives

We've presented a model for compositional work based on the transformation of previously existing material. This model is made possible by the segmentation of data, and the triggering of external transformations controlled by user chosen Walsh functions, which have been shown to be much convenient for such task. In practice, the symmetrical disposition of values of a Walsh function implies that operations of transformation are performed symmetrically in time. However, one should not count on recognizing such a symmetry in the output, since is unlikely for the input to share similar symmetrical properties. The processes here described have been compositionally used in *Piano Solo IV* [4] by Patricio da Silvato, a set of Walsh variations on a theme by J.S. Bach.

Our model can be generalized to include other functions besides the Walsh ones. The next step will be to use the so called *Sequency Functions* [7, 6] (an obvious generalization of the Walsh Functions)as triggers, and to incorporate other music parameters, such as rhythm, instrumentation, dynamics, etc. This will be accomplished elsewhere.

5. APPENDIX: WALSH FUNCTIONS AND HADAMARD MATRICES

Walsh functions became important for representation of signals through the superposition of members of a set of simple functions which are easy to generate and define [6, 7]. They were first used in Computer Music as a device to control Sound Synthesis [8]. They form an ordered set of rectangular waveforms taking only two amplitudes values $+1$ and -1. A simple example of a set of rectangular waveforms are the Rademacher Functions which can be defined as

$$RAD(n,t) = sign[sin(2^n \pi t)] \tag{1}$$

where $0 \leq t \leq 1$. Rademacher functions have two arguments n and t such that $RAD(n,t)$ has 2^{n-1} periods of square wave over a normalised time base, or interval $[0,1]$.

The problem with Rademacher System is that it is not complete in the sense that any signal can be decomposed, like in a Fourier Series, as a sum (perhaps infinite) of Rademacher Functions. The simplest complete set of rectangular functions (waveforms in the context of this work) is the Walsh Set. From the point of view of signal representation, Walsh functions consist of trains of square pulses (with the allowed states being -1 and 1) such that transitions may only occur at fixed intervals of a unit time step, the initial state is always 1. In general Walsh functions are defined in a Time Base interval T and periodically extended for intervals of length $kT, k \in \mathbf{Z}$. They are completely defined by two parameters, its *sequency order* n and its time variable t. It is denoted $WAL\,(n,t)$, with $n = 0, 1, 2, \ldots, N-1$, and N is the order of Walsh Functions defined below. Using a normalized time variable t/T the Walsh functions can be defined in the interval $[0,1]$. In addition, they are symmetrical about the centre and so, when they are defined in the interval $[-1/2, 1/2]$ they are symmetrical. The even functions are collectively named CAL and the odd ones SAL which are in certain sense the counterparts of the cosine and sine trigonometric functions. so, we can write

$$WAL\,(2k,t) = CAL\,(k,t)\,, k = 1, 2, \ldots, N/2 \tag{2a}$$
$$WAL\,(2k-1,t) = SAL\,(k,t)\,, k = 1, 2, \ldots, N/2 \tag{2b}$$

Both Rademacher and Walsh Sets are orthogonal systems in the same way as Fourier Systems of sin and cos functions. Other systems of rectangular functions do exist, such as Haar and Slant Functions. See reference [6] for more information and bibliography on this subject.

Walsh functions can be ordered in a number of ways. One of them is the so called sequency order. The sequency k of a Walsh function is defined as half the number of zero crossings in one cycle of the time base. Walsh functions with nonidentical sequencies are orthogonal and the product of two Walsh functions is also a Walsh function.

A way to generate Walsh functions is through the so called *Hadamard Matrix* [7] when arranged in the "sequency" order. A Hadamard matrix of order N is a type of square matrix whose entries are only $+1$ and -1 and such that

$$\mathbf{H}\mathbf{H}^T = N\mathbf{I} \tag{3}$$

This equation implies that the rows of $\mathbf{H}$ (or the Walsh functions of order N) are orthogonal. In the so called *normal form* the first row and the first column are formed only by $+1$. The lowest-order Hadamard matrix is the 2 dimensional matrix

$$\mathbf{H}_2 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4}$$

An important theorem on Hadamard Matrices is stated as:

Theorem: If $\mathbf{H_m}$ and $\mathbf{H_m}$ are matrices of orders m and n respectively, then their Direct Product is an $\mathbf{H}$ matrix of order mn. The proof of this theorem as well the construction of some Hadamard matrices for some particular values of their dimensions can be founded in [7].

Most of constructions of Hadamard Matrices are based on Direct (Kronecker) Product of two matrices. The definition of Direct Product as as follows. If $\mathbf{A} = (a_{ij})$ is an $m \times m$ matrix and $\mathbf{B} = (b_{rs})$ is an $m \times m$ matrix, then the Direct Product $\mathbf{A} \otimes \mathbf{B}$ is the $mn \times mn$ matrix given by

$$\mathbf{A} \otimes \mathbf{B} \begin{bmatrix} a_{11}B & a_{12}B & \ldots & a_{1m}B \\ a_{21}B & a_{22}B & \ldots & a_{2m}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{i1}B & a_{i2}B & \ldots & a_{im}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}B & a_{m2}B & \ldots & a_{mm}B \end{bmatrix} \tag{5}$$

Using this theorem, higher-order matrices, with dimension 2^n are easily obtained by the recursive relationship

$$\mathbf{H}_N = \mathbf{H}_{N/2} \otimes \mathbf{H}_2 \tag{6}$$

where $N = 2^n$. Thus, for example,

$$\mathbf{H}_4 = \mathbf{H}_2 \otimes \mathbf{H}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{7}$$

All the values at a line of a Hadamard Matrix are the values of a particular Walsh Function. Clearly the user has the freedom to map several kinds of mathematical objects to these values. In this work we map time intervals, which are obtained by segmentation, to them.

The ordering of Walsh Functions in a Hadamard Matrix is named *natural ordering* and in this case they are also named Hadamard Functions and denoted by $HAD(k, t)$, where $k = 0, 1, \ldots, N$ and the variable t is interpreted in our work as time. For example, the Walsh Functions of order 8 ($\mathbf{H}_8$) are given by the matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{8}$$

from which we can read, for example, the Walsh function

$$HAD\,(5, t) = [1, -1, 1, -1, -1, 1, -1, 1] \tag{9}$$

6. ACKNOWLEDGMENTS

This work was supported by Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP), Coordenadoria de Aperfeiçoamento de Pessoal de Ensino Superior (CAPES), Brazil, and Fundação da Ciencia e Tecnologia (FCT), Lisbon, Portugal. The authors are grateful to Eduardo Coutinho for suggestions and figures.

7. REFERENCES

[1] P. da Silva, *David Cope and the Experiments in Music Intelligence*, http://www.spectrumpress.com/da-silva-web-papers.html, (2003).

[2] P. da Silva, *Representation of Music in Cellular Space*, Los Angeles, CA: Spectrum Press, (2005).

[3] P. da Silva, *Cellular Variations*, Los Angeles, CA: Spectrum Press, (2005).

[4] P. da Silva, *Piano Solos I - IV*, Los Angeles, CA: Spectrum Press, (2005).

[5] D. Cope, *Computers and Musical Style*, Madison, Wisconsin: A-R Editions Inc, (1991).

[6] K. G. Beauchamp, *Walsh Functions and Their Applications*, London, Academic Press, (1975).

[7] M. Hall Jr.,*Combinatorial Theory*, 2^{nd} Edition, John Wiley and Sons Inc., (1986).

[8] M. Rozenberg, *Microcomputer-controlled sound processing using Walsh Functions*, Computer Music Journal, Vol. 3, No 1, pp. 42-47, (1979).